OPTIMAL IRRIGATION PLANNING AND OPERATION OF RESERVOIR USING SELF-ADAPTIVE CUCKOO SEARCH ALGORITHM (SACSA)

Sriman Pankaj Boindala¹, Vasan Arunachalam²

ABSTRACT

Irrigation planning and operation of reservoir is one of the popular applications of optimization in water resource management. The complexity of this problem lies in its large dimensions (variables) and multiple constraints. This study focusses on determining the optimal cropping pattern which would maximize the annual net benefits from crop production. Cuckoo search is one of the promising Swarm Intelligence (SI) based metaheuristic algorithms for solving various engineering problems. There are two main parameters which govern the convergence speed and accuracy, (i) step length (α) and (ii) switching parameter (pa). It is a tedious task to find out the most suitable values of these two parameters which can efficiently solve a particular problem. To overcome this problem a self-adaptive version of this cuckoo search algorithm is proposed in this work. An Indian case study Mahi Bajaj Sagar project, Rajasthan, India is taken as an example for reservoir optimization problem. The results show that the optimal cropping pattern is obtained for a smaller number of function evaluations using SACSA making it much better suitable algorithm for solving this complex water resource problem. The improvement in the performance of SACSA than the original cuckoo search is due to its problem adaptive nature.

Keywords: Cuckoo Search, Swarm Intelligence, Self-adaptive Cuckoo Search, water resource management, reservoir optimization.

1. INTRODUCTION

With the increase in global population water supply systems are being stressed to satisfy the production demands of the country. Water scarcity has become an important constraint on economic development. This scarcity has led to competition between various economic sectors like agriculture and power generation that rely on water resources (Debnath et al. 2009). Reservoir operations directly affect various economic sectors, so optimization of these operations to maximize the new benefits by satisfying the need of these sectors has been a critical issue (Shreedhar et al. 2015). Past researchers have focused on obtaining the best cropping pattern to maximize the benefits from agriculture and power supply by optimizing the water resources using linear programming, dynamic programming and other traditional optimization algorithms (Shreedhar et al. 2015). With the invent of latest optimization algorithms based on evolutionary theory and swarm intelligence, researchers started to use these nontraditional methods to find the optimum cropping patterns (Adeyemo and Stretch 2018; Memmah et al. 2015). Most of these nontraditional optimization algorithm’s performances are governed by algorithm specific parameters. Choosing the appropriate parameter for a given problem is always a challenging task for an individual. To overcome this situation, research has moved towards developing self-adaptive algorithms (Mareli and Twala 2018; Naik and Panda 2016; Nobakhti and Wang 2008; Qin and Suganthan 2005). The current study focusses on a self-adaptive mechanism which changes the values of these two parameters based on the fitness

¹ Graduate student, Department of Civil Engineering, Birla Institute of Technology and Science E-mail: srimanpankaj@gmail.com
² Professor, Department of Civil Engineering, Birla Institute of Technology and Science E-mail: vasan@hyderabad.bits-pilani.ac.in
and the iteration number. The mechanism is explained in methodology section. This developed algorithm’s performance is verified by test function and its proven to be better than the original optimization algorithm. This new algorithm is then applied to solve the reservoir optimization problem.

2. SELF-ADAPTIVE CUCKOO SEARCH ALGORITHM (SACSA):

Cuckoo search is one of the latest nature inspired meta-heuristic algorithm developed by Xin-She-Yang in 2009 (Yang and Deb 2009). It uses a levy-flight based search walk instead of normal random walk. Comparative analysis of this algorithm with other meta-heuristic algorithms like PSO, FA, showed that CS algorithm is performing better (Civicioglu and Besdok 2013). Most of the engineering-based problems like spring design and welded beam design are solved by using cuckoo search algorithm. Cuckoo search has also proven to be working well in water resources sector too (Mohammadrezapour et al. 2017; Wang et al. 2012; Yasar 2016). The performance of this algorithm is governed by two main parameters (i.e) Step Size control parameter alpha ‘α’ and shifting parameter ‘pa’ (Yang 2014). These two parameters govern the exploration and exploitation ability of the algorithm (Yang and Deb 2009). These parameters are essential for optimizing the performance of the algorithm and the values of these parameters vary with the type of problem. Choosing the appropriate parameter for obtaining the ideal solution with least computation power is a difficult task. To solve this problem, many researchers focused on developing self-adaptive versions of these algorithms. Self-adaptive algorithms, vary their parameters based on either fitness or iteration number or randomization or a combination of these. The current study is focused on evaluating one such self-adaptive version of cuckoo search developed for obtaining ideal crop patterns. The self-adaptive cuckoo search algorithm follows the below pseudo code.

3. PSEUDO CODE FOR SELF-ADAPTIVE CUCKOO SEARCH ALGORITHM (SACAS)

Step 1: formulate the Objective function \( f(n) \), \( n=(x_1,x_2,x_3,x_4,\ldots,x_d) \) where \( f \) is the function and \( n \) is the variable set (nest) and \( d \) is the dimension of the problem’s search space.

Step 2: decide upon a number of nests (NN) to be considered (in general 10-20 times the number of dimensions).

Step 3: assume alpha for all the nests as 1 initially. (alpha = ones (1, NN);)

Step 4: generate the values for variables within the bounds for all the nests.

Step 5: evaluate all the nests and find the ideal nest, fitness values of each nest, fitness of ideal and anti-ideal nests.

Step 6: use this ideal nest and corresponding alpha to guide the levy flight step size.

Step 7: evaluate the new nests produced in step 5

Step 8: update the best nest

Step 9: calculate \( P \) based on the equation 2.

Step 10: replace the nests with a probability \( P \). \( P \sim (0-1) \).

Step 11: evaluate all the nests and update the best nest.

Step 12: update the alpha by the equation 1.

Step 13: repeat the steps 5, 6, 7, 8, 9, 10, 11, 12 iteratively.
Step 11: After several iterations, the nests converge to a particular solution.

\[ \alpha (t) = \left( \frac{1}{t} \right)^{\frac{F_{\text{ideal}} - F_t}{F_{\text{anti-ideal}}}} \]  
Here \( \alpha \) - alpha, \( t \) - nest number, \( t \) - iteration number, \( F_{\text{ideal}} \) - Fitness value of ideal nest, \( F_{\text{anti-ideal}} \) - Fitness value of anti-ideal nest

\[ P(t) = P_{\text{max}} \times e^{t/time} \]  
Here \( P \) – Shifting parameter, \( P_{\text{max}} \) – Maximum value of the shifting parameter, \( t \) – iteration number, \( \text{time} \) – maximum number of iterations

The alpha parameter governs the step length of the updated nest, the alpha will be decreasing with improvement in fitness and iteration from 1. Equation 1 ensures the step length to vary for each nest based on its relative distance from the ideal and anti-ideal solutions. The P- Shifting parameter ensures that algorithm does not go into a local optimum. Mareli and Twala, 2018 (Mareli and Twala 2018) studied the convergence rate with linear, exponential, cubic adaptive mechanism and concluded that exponential mechanism is performing better than other mechanism. The efficiency of the self-adaptive algorithm is evaluated by various test functions. Five test functions with 10 dimensions are tested for 50 times with a tolerance convergence criterion of 10 The maximum and minimum function evaluations needed for satisfying the termination criteria among all the 50 runs are mentioned below for cuckoo search and self-adaptive cuckoo search. The number of nests is taken as 25 for both the algorithms.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Function evaluations</td>
<td>Function evaluations</td>
</tr>
<tr>
<td>Ackley</td>
<td>426</td>
<td>484</td>
</tr>
<tr>
<td>Dixon-Prince</td>
<td>512</td>
<td>2999</td>
</tr>
<tr>
<td>Griewank</td>
<td>2318</td>
<td>7481</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>1421</td>
<td>3780</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>991</td>
<td>1331</td>
</tr>
</tbody>
</table>

4. CASE STUDY

In this present study, this new self-adaptive algorithm-based irrigation planning model is formulated and applied to a case study of Mahi Bajaj project, Banswara district, Rajasthan, India, to evolve a suitable optimum cropping pattern to yield maximum net benefits while meeting all the water supply demands involved with the project. This water resources project is associated with three main canals (Left main canal, Right main canal, Bhungra canal) and two hydroelectric power houses. The map of this project is shown in figure 1. Vasan et al,2005 and Trivedi et al, 2006 worked on obtaining the cropping patterns using Real coded genetic algorithm, simulated annealing, simulated quenching, differential evolution and stochastic linear programing. The complete details of the project are mentioned in table 1.
Table 2. Reservoir salient features (Source: MBSP Report on Project Estimate of Unit-II, 2001)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Reservoir Level</td>
<td>EL 281.50 m</td>
</tr>
<tr>
<td>Water Spread Area</td>
<td>142.90 Km²</td>
</tr>
<tr>
<td>Live Storage Capacity</td>
<td>1829.27 Mm³</td>
</tr>
<tr>
<td>Dead Storage Capacity</td>
<td>351.13 Mm³</td>
</tr>
<tr>
<td>Sharing of Water</td>
<td></td>
</tr>
<tr>
<td>Madhya Pradesh</td>
<td>368.11 Mm³</td>
</tr>
<tr>
<td>Gujarat State</td>
<td>1132.67 Mm³</td>
</tr>
<tr>
<td>Culturable Command Area (CCA)</td>
<td>1,23,500 ha</td>
</tr>
</tbody>
</table>

5. OBJECTIVE FUNCTION

\[
\text{Annual}_\text{NetBen} = \sum_{i=1}^{36} B_i + A_i - \sum_{j=1}^{12} \text{GW}_j
\]

A(1-36) = Cropping areas of 12 types of crops grown in three regions (12x3 = 36), B(1-36) = Benefits obtained from 12 crops grown in three regions (12x3=36), \( \text{GW}_j \) = cost incurred for extraction of unit ground water, \( \text{GW}(1-12) \) = Ground water extracted in 12 months

5.1 Constraints

Continuity equation at the main reservoir

\[
S_{t+1} = S_t + I_t - \text{IDB}_t - \text{PH}_t - \text{EV}_t - \text{USMP}_t
\]

S - Storage, \( t \) - current month, IDB - irrigation demand for Bhungra canal, USMP - Up stream requirement for Madhya Pradesh, I - inflow, \( \alpha \) - Dependability of inflow.
Continuity Equation at Kagdi Pick Up Weir

\[ PH_{1t} + GW_{2t} = IDL_t + IDR_t + PH_{2t} \] (5)

IDL = Irrigation demand for left main canal, IDR = irrigation demand for right main canal, GW = ground water, \( t \) (1-12) – months

Command Area Limitations

Kharif Season (LMC)

\[ \sum_{i=1}^{6} A_i \leq p_{ki} \times CCA_L \] (6)

Rabi Season (LMC)

\[ \sum_{i=7}^{12} A_i \leq p_{ri} \times CCA_L \] (7)

Kharif Season (RMC)

\[ \sum_{i=13}^{18} A_i \leq p_{ki} \times CCA_R \] (8)

Rabi Season (RMC)

\[ \sum_{i=19}^{24} A_i \leq p_{ri} \times CCA_R \] (9)

Kharif Season (BC)

\[ \sum_{i=25}^{30} A_i \leq p_{ki} \times CCA_B \] (10)

Rabi Season (BC)

\[ \sum_{i=31}^{36} A_i \leq p_{ri} \times CCA_B \] (11)

\( A_i \) – area of crop ‘i’, \( P_{ki} \) - Percentage intensity of crop ‘i’ in kharif season, \( P_{ri} \) – Percentage intensity of crop ‘i’ rabi season, CCA\(_L\), CCA\(_R\), CCA\(_B\) = Culturable command area under Left Main canal (ha), Right Main canal (ha) and Bhungra canal (ha) respectively.

Crop Diversion Requirements

\[ \sum_{i=1}^{12} CWR_{it} A_i - IDL_t = 0 ; t = 1 - 12 \] (12)

\[ \sum_{i=13}^{24} CWR_{it} A_i - IDR_t = 0 ; t = 1 - 12 \] (13)

\[ \sum_{i=25}^{36} CWR_{it} A_i - IDB_t = 0 ; t = 1 - 12 \] (14)

CWR\(_{it}\) – Crop water requirement of crop ‘i’ during month ‘t’.

Canal Capacity Restrictions

\[ PH_{2t} + IDL_t \leq CCL_t ; t = 1 - 12 \] (15)

\[ IDR_t \leq CCR_t ; t = 1 - 12 \] (16)

\[ IDB_t \leq CCB_t ; t = 1 - 12 \] (17)

CCL, CCR, CCB = Canal Capacity of Left, Right and Bhungra canal respectively. PH\(_{2t}\) - Power house-2 demand for month ‘t’.

Water Requirements (WR) for Hydropower Generation

\[ PH_{1t} \geq WR_{1t} \] (18)

\[ PH_{2t} \geq WR_{2t} \] (19)
Minimum and Maximum Areas of Crops

\[ A_{\text{MAX}}_i \leq A_i \leq A_{\text{MAX}}; \ i = 1, ..., 36 \]  \hspace{1cm} (20)

Ground Water Withdrawals

\[ \sum_{i=1}^{12} G_W_i \leq T_G W \]  \hspace{1cm} (21)

TGW – Total Ground Water allowed to extract.

Live Storage (LS) Restrictions

\[ S_t \leq L_S; \ t = 1, ..., 12 \]  \hspace{1cm} (22)

Table 3. Variation of inflows with dependability

<table>
<thead>
<tr>
<th>Inflows (Tmc)/Month</th>
<th>60 %</th>
<th>65 %</th>
<th>70 %</th>
<th>75 %</th>
<th>80 %</th>
<th>85 %</th>
<th>90 %</th>
</tr>
</thead>
<tbody>
<tr>
<td>July</td>
<td>180.0665</td>
<td>105.8994</td>
<td>79.0891</td>
<td>73.256</td>
<td>61.8722</td>
<td>59.707</td>
<td>31.658</td>
</tr>
<tr>
<td>August</td>
<td>1176.591</td>
<td>1037.045</td>
<td>885.0574</td>
<td>73.256</td>
<td>61.8722</td>
<td>338.2725</td>
<td>331.3313</td>
</tr>
<tr>
<td>September</td>
<td>1527.21</td>
<td>1396.6</td>
<td>1233.257</td>
<td>1066.694</td>
<td>926.2425</td>
<td>493.3394</td>
<td>343.511</td>
</tr>
<tr>
<td>October</td>
<td>1661.27</td>
<td>1560.76</td>
<td>1339.209</td>
<td>1057.123</td>
<td>804.6511</td>
<td>598.8552</td>
<td>486.143</td>
</tr>
<tr>
<td>Total</td>
<td>4569.235</td>
<td>4120.398</td>
<td>3554.022</td>
<td>2880.92</td>
<td>2145.45</td>
<td>1497.32</td>
<td>1160.37</td>
</tr>
</tbody>
</table>

6. RESULTS

After taking the objective function and all the constraints, the total number of variables is 120, and 79 constraints making this a high dimensional linear optimization problem. The problem is solved using 60% dependable inflow to 85% dependable inflows. The inflow dependabilities are calculated using Weibull distribution. The requirements are not met for 85% dependable inflow. There was additional need of surface/ground water to satisfy the irrigation requirements which made it infeasible. The maximum net benefit obtained with varying inflow reliability is tabulated in table-3. The cropping pattern for 80% dependable inflows is mentioned in the bar graph below (figure -3). The maximum benefits are obtained when most of the crops to be grown in Bhungra canal are Zaid.
crops. The cropping patterns of both right and left canal are almost similarly distributed. Convergence graph obtained for maximization of annual net benefits is shown in figure-2.

### Table 4. Maximum benefits for various inflow reliabilities

<table>
<thead>
<tr>
<th>Inflows (reliability)</th>
<th>60 %</th>
<th>65 %</th>
<th>70 %</th>
<th>75 %</th>
<th>80 %</th>
<th>85 %</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-Best (maximum benefits in million rupees)</td>
<td>₹1,829.95</td>
<td>₹1,828.68</td>
<td>₹1,823.60</td>
<td>₹1,812.45</td>
<td>₹1,777.30</td>
<td>Not feasible</td>
</tr>
</tbody>
</table>

**Figure 3.** Crop patterns of three canals with 80% reliable inflows

#### CONCLUSIONS

The proposed algorithm is working better than original cuckoo search version, this can be concluded from statistics mentioned in Table 1. The developed SACSA has been successfully applied to the real-life case study in this work to obtain best cropping patterns.

The study reviews the applicability of self-adaptive cuckoo search in the field of reservoir optimization. The above-mentioned algorithm is found to be working well and the results are similar to the results obtained using other algorithms by (Vasan et al, 2005).

The self-adaptive mechanism enables the user to use the algorithm without any concern about the tuning parameters. The sensitivity analysis of the algorithm parameters can be eliminated by using this technique.

Even with 80% dependable inflows, the reservoir can perform efficiently and can provide an annual net benefit of 1770.3 million rupees. With the cropping pattern obtained from the algorithm.
8. REFERENCES


